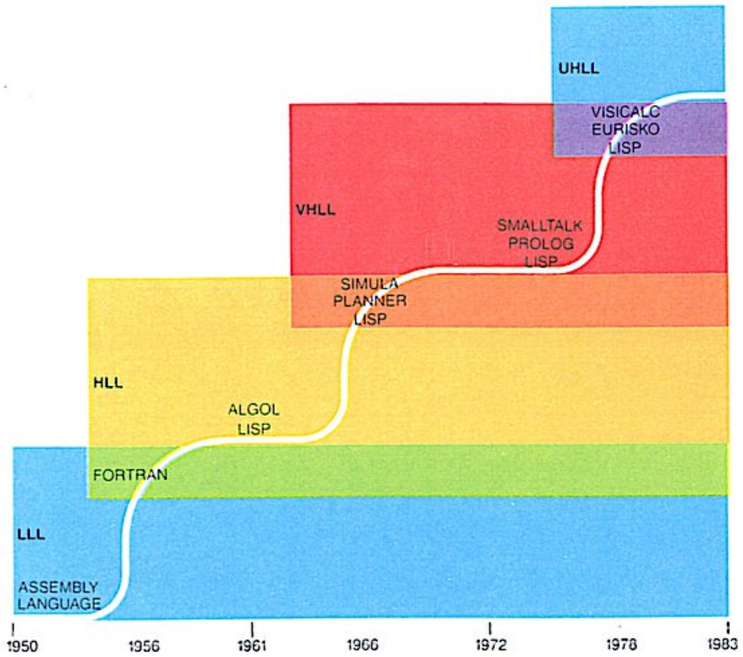
OaaS

Objects as a Service

$1T

Global business software and services market size by 2029[†]

Software development technologies aren't progressing since 1980s



SOFTWARE GENRES succeed one another at sporadic intervals, as is shown here through the example of some programming languages. Languages are categorized rather arbitrarily by level, although the levels (*colored bands*) overlap. There are low-level languages (LLL), high-level languages (HLL), very-high-level languages (VHLL) and ultrahigh-level languages (UHLL). In the evolution of programming languages a genre is established (*horizontal white lines*), then after a few years an improvement is made (*curved white lines*). In time the improved language is seen to be not merely a "better old thing" but an "almost new thing," and it leads to the next stable genre. The language Lisp has changed repeatedly, each time becoming a new genre.

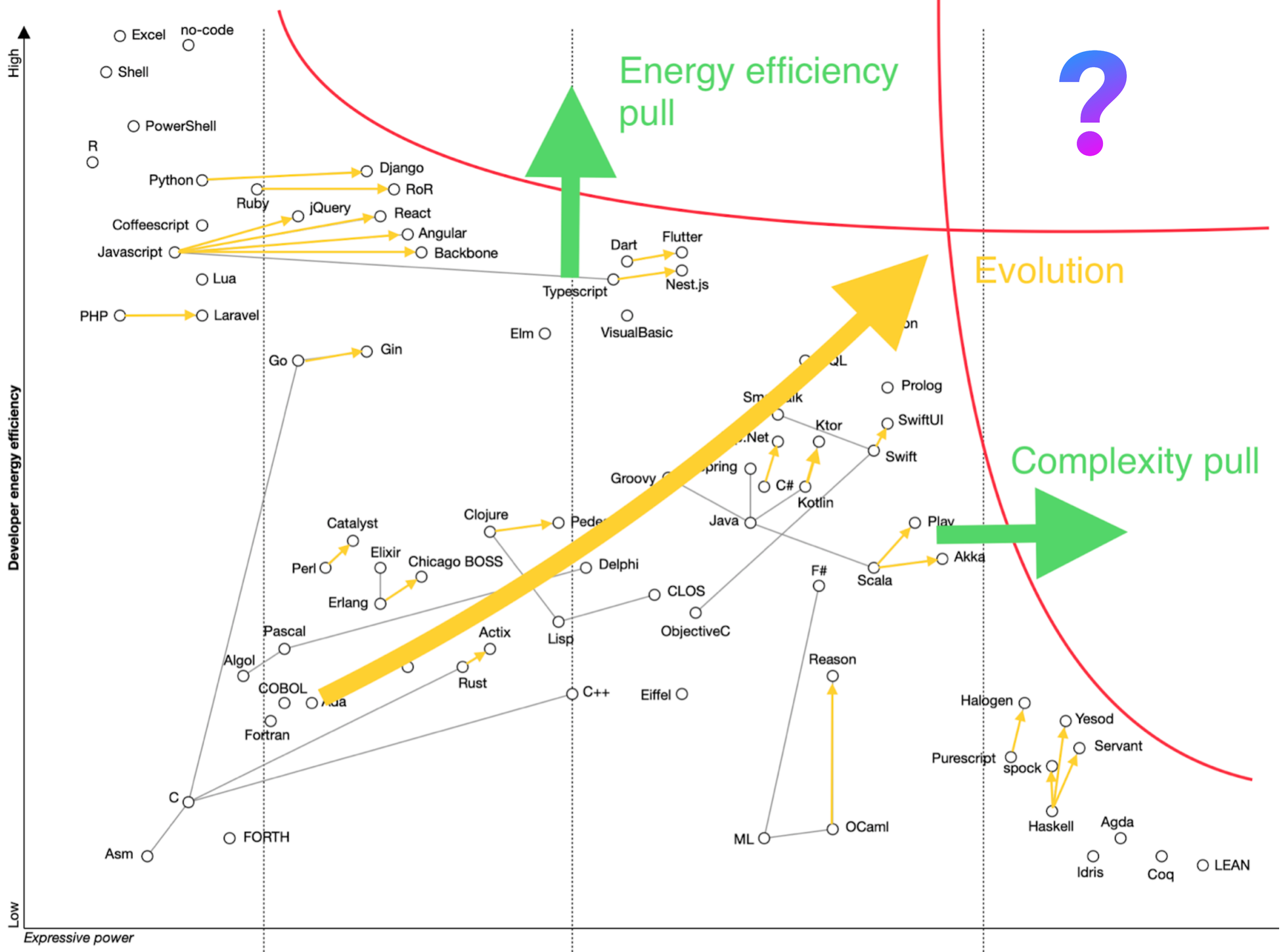
Current generation tools reached universality, by horizontally scaling with humans instead of further scaling tech vertically up, and stopped evolving upwards.

The *no-code* paradigm in its current incarnation does not offer qualitative next-level capabilities. Rather, it is providing vertical solutions, which are not flexible and not powerful enough even for those vertical tasks.

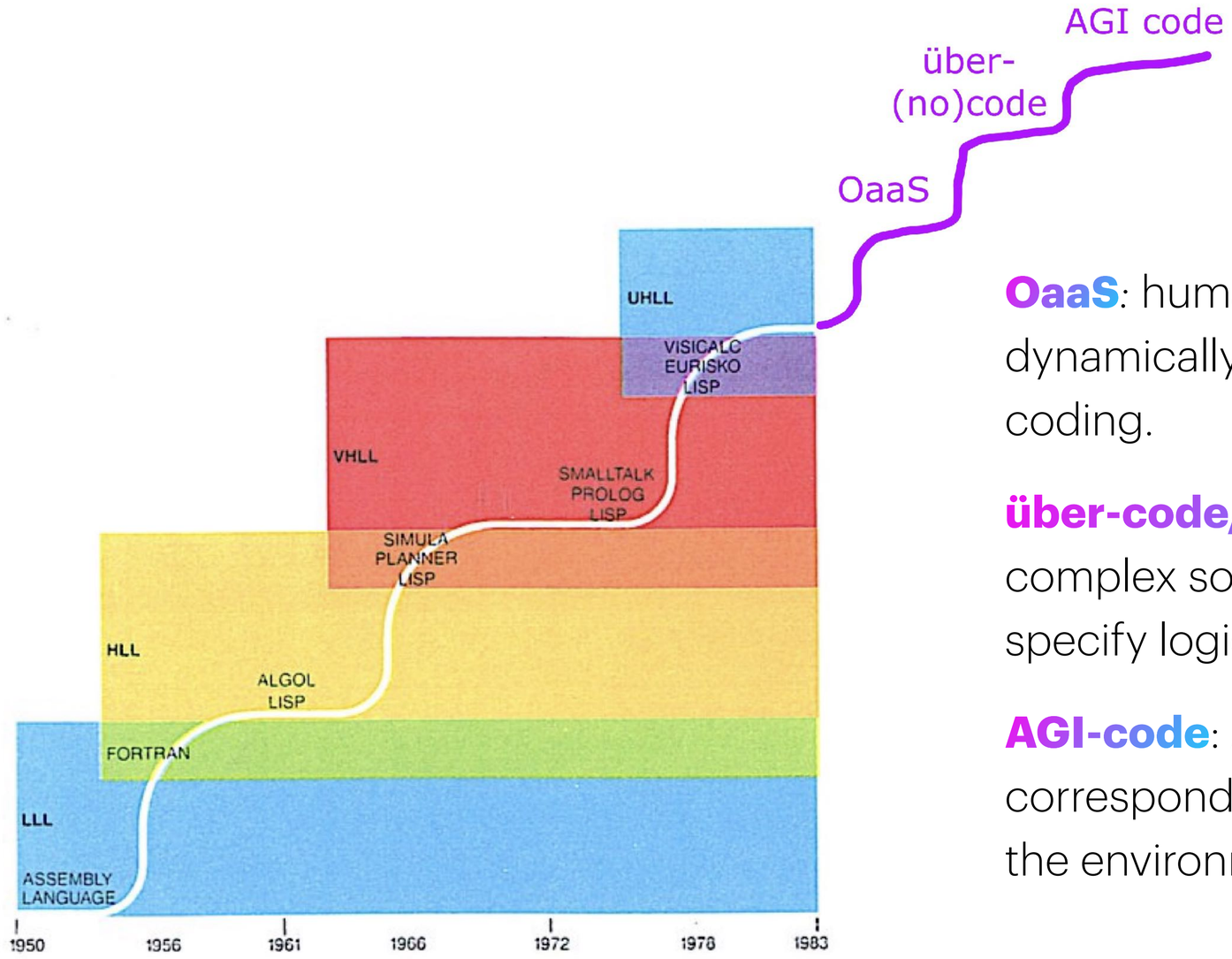
www.quora.com/What-was-the-last-breakthrough-in-computer-programming/answer/Alan-Kay-11

Evolution model

traversable.space/#essays/guts-d



Future of Software Development



OaaS: humans build complex software *systems* out of dynamically created and composed objects, without coding.

über-code/über-nocode: humans+*AI assistants* build complex software systems using human language to specify logical models, constraints and attractors.

AGI-code: AGI builds and updates logical models and the corresponding software by interacting with humans and the environment.

Why OaaS

Current software is built with *Algorithms&Data structures* programming model. Its constructs are not powerful enough to model evolving systems. It only works well for **simple systems**. Such systems are rigid, non-flexible, fragile, do not scale to evolution.

Industry solutions to this problem create exponentially more **own complexity** than the actual business functionality. Building software is more like building an Egyptian pyramid, rather than evolving an organism.

Real-world business applications, evolving in time and scale, and interacting with the external environment, quickly become **complex systems**. This is where the **impedance mismatch** arises.

We need a "*organizing complex systems*" model

"Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other, with no structural integrity, but just done by brute force and thousands of slaves."

— Alan Kay.

Objects as a Service FTW!

OaaS provides an environment and a UI which let users specify:

- ***business entities,***
- ***business processes,***
- ***connections,***
- ***integrations,***
- ***constraints and required states,***
- ***business rules.***

Users build software functionality step-by-step, by creating new objects with no-code tools, and reusing existing ones.

Objects communicate and form flexible dynamic workflows, networks, feedback loops, recombine data and UIs. Coherent evolvable business applications emerge.

OaaS manages the implementation details:

infrastructure, scalability, persistence, data layer, messaging, protocols, evaluation strategies, dynamic dispatch, AAA, logging, metrics, resilience, availability, UI composition, testing, integrations, messaging, routing, deployment, hosting, CI/CD, version control,

backup and recovery, environment, sessions, contexts, state machines, distributed transactions, IDE, programmatic UI, SAT constraints solver, configuration, state history and rollback, etc.

Next level of expressive power

Objects live in the **cloud**, are universally addressed, communicate natively.

API integrations are just **natural** objects communication.

Objects can be **reused** from an inventory, can be bought and sold on a **marketplace**.

This is the **next step** after DevOps, Kubernetes, containers, OS, IaaS/PaaS/FaaS/SaaS.

Cloud Functions, the current pinnacle of cloud design, is not a powerful enough abstraction to **simulate business entities** and relations. Objects are.

This is an extension of ideas at PARC, and later of Steve Jobs, redesigned for **21st century** and cloud.

There was no cloud and demand back then.

We do **have cloud and demand now**.

Next level in user energy efficiency

- O.G. OOP × no-code × cloud × zero maintenance × Internet-wide scope
- building and operating software the way **the Internet** is built and operated: it never stops, but always evolves and scales
- a medium for sculpting software to business requirements, **in dynamics**
- for **business domain experts**, not programmers
- for **programmers**, too
- if children can do it **anyone** can too

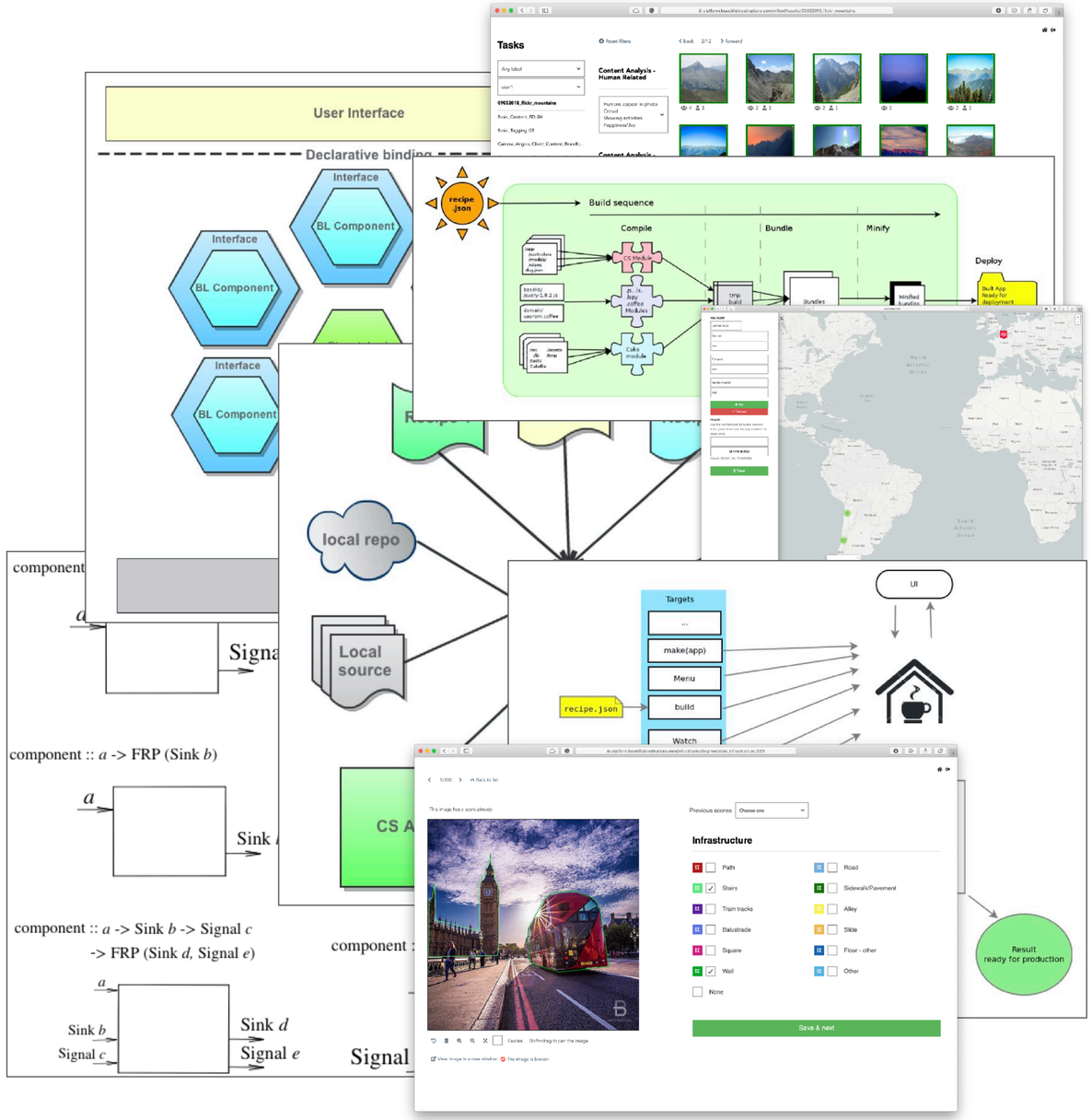
"The best way to predict the future is to invent it."

— Alan Kay.

Proven with prototypes

OaaS model **has been implemented** in several prototypes, from user UI and workflows to core engine.

These prototypes have been **used in real-world production applications**, and did provide the proverbial 10x results both from the expressive power and human energy efficiency points of view.



Customer segments

- people who are **experts** in their domains and know the logical model of the functionality they need, but are not software developers (SMB owners, online sellers, creators etc)
- enterprise **architects** who need to integrate many moving pieces into a working and evolvable solution
- professional software **developers** who want to build powerful and flexible solutions for their clients in less time and efforts
- **startups** and other product teams who want to minimize time to market
- school **students** for making software for their classes :-)
- many others

Everyone able to step-by-step build a logical domain model is able to build software.

The industry is expanded by 10sX

Value is created

“Instead of innovating out from the present, what you want to do is invent the future from the future. Go out and live in the future and bring the future back.”

“Computer science inverts the normal. In normal science, you're given a world, and your job is to find out the rules. In computer science, you give the computer the rules, and it creates the world.”

“If you want to make money, don’t make a startup, start an industry.”

— Alan Kay